Mean Field Game Guided Deep Reinforcement Learning for Task Placement in Cooperative Multiaccess Edge Computing

Dian Shi, *Student Member, IEEE*, Hao Gao, *Student Member, IEEE*, Li Wang, *Senior Member, IEEE*, Miao Pan, *Senior Member, IEEE*, Zhu Han, *Fellow, IEEE*, and H. Vincent Poor, *Life Fellow, IEEE*

***Abstract*—Cooperative multiaccess edge computing (MEC) is a promising paradigm for the next-generation mobile networks. However, when the number of users explodes, the computational complexity of the existing optimization or learning-based task placement approaches in the cooperative MEC can increase significantly, which leads to intolerable MEC decision-making delay. In this article, we propose a mean field game (MFG) guided deep reinforcement learning (DRL) approach for the task placement in the cooperative MEC, which can help servers make timely task placement decisions, and significantly reduce average service delay. Instead of applying MFG or DRL separately, we jointly leverage MFG and DRL for task placement, and let the equilibrium of MFG guide the learning directions of DRL. We also ensure that the MFG and DRL approaches are consistent with the same goal. Specifically, we novelly define a mean field guided Q-value (MFG-Q), which is an estimation of the Q-value with the Nash equilibrium gained by MFG. We evaluate the proposed method's performance using real-world user distribution. Through extensive simulations, we show that the proposed scheme is effective in making timely decisions and reducing the average service delay. Besides, the convergence rates of our proposed method outperform the pure DR-based approaches.**

***Index Terms*—Deep reinforcement learning (DRL), mean field game (MFG), multiaccess edge computing (MEC), task placement.**

I. Introduction

IN RECENT years, with the continuous prosperity of the mobile Internet industry and Internet of Things, the advance of widely used mobile terminals spurs the development of smart and media applications with high CPU power requirements, such as live video, virtual reality, and intelligent image recognition. Those CPU-hungry applications further promote the emergence of multiaccess edge computing (MEC) [1], [2] architecture, which deploys cloud-like computing servers, called MEC servers, at the edge of the cellular networks. This MEC architecture, which pushes computational capabilities closer to end users, is a promising solution to the long latency and backhaul bandwidth limitation problem.

Due to the heterogeneity of resource availability among MEC servers, the cooperative MEC architecture is proposed to take the excellent usage of the computational resources. Through scheduling to do tasks uploaded by end users among multiple MEC servers, the integration of service delay and power consumption for the system can potentially be reduced, while how to appropriately place the tasks poses significant challenges. Especially, when the number of end users grows explosively, making timely task placement decisions is very difficult.

In the literature, many efforts have been devoted to the task placement in the cooperative MEC, and most of them are optimization-based methods. Although they can make proper task placement decisions, their approaches have many critical limitations, e.g., computational complexity explodes with the number of users increasing, the optimization needs to be resolved whenever the distribution of the new task is coming, etc. These deficiencies may lead to the intolerable MEC decision-making delay. Recently, many scholars apply the reinforcement learning (RL) approach in the cooperative MEC. RL [3], as a type of the machine learning method, has been widely applied in many applications of artificial intelligence, e.g., Alpha Go [4], playing video games [5], scheduling TNC vehicles [6], [7], and solving communication problems [8]. As for RL, the agent constantly interacts with the learning environment and makes decisions according to the Markov

Manuscript received December 17, 2019; revised February 29, 2020; accepted March 13, 2020. Date of publication March 27, 2020; date of current version October 9, 2020. The work of Dian Shi and Miao Pan was supported by the U.S. National Science Foundation under Grant CNS-1350230 (CAREER), Grant CNS-1646607, Grant CNS-1702850, and Grant CNS-1801925. The work of Hao Gao and Zhu Han was supported in part by U.S. Multidisciplinary Research Program of the University Research Initiative AFOSR under Grant MURI 18RT0073, and in part by NSF under Grant EARS-1839818, Grant CNS1717454, Grant CNS-1731424, and Grant CNS-1702850. The work of Li Wang was supported in part by the National Natural Science Foundation of China under Grant 61871416, in part by the Fundamental Research Funds for the Central Universities under Grant 2018XKJC03, and in part by the Beijing Municipal Natural Science Foundation under Grant L192030. The work of H. Vincent Poor was supported in part by the U.S. Air Force Office of Scientific Research under Grant MURI FA9550-18-1-0502. *(Corresponding author: Miao Pan.)*

Dian Shi, Hao Gao, and Miao Pan are with the Electrical and Computer Engineering Department, University of Houston, Houston, TX 77004 USA (e-mail: dshi3@uh.edu; hgao5@uh.edu; mpan2@uh.edu).

Li Wang is with the School of Electronic Engineering, Beijing University of Post and Telecommunications, Beijing 100876, China (e-mail: liwang@bupt.edu.cn).

Zhu Han is with the Electrical and Computer Engineering Department, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 446-701, South Korea (e-mail: zhan2@uh.edu).

H. Vincent Poor is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: poor@princeton.edu).

Digital Object Identifier 10.1109/JIOT.2020.2983741

process. Leveraging the RL approach can eliminate the restriction on dynamic changes for task distribution, but the learning speed and convergence rates, which have a great impact on decision-making delay, are non-negligible for task placement in the MEC. Meanwhile, a relatively new concept in the game theory domain, called mean field game (MFG) [9], has been applied in many engineering problems [10], [11]. It models the game as one player interacting with the collective behavior of all other players, and provides equilibrium solutions. Thus, MFG is typically fit for a large number of players' scenario, such as multiple tasks scheduling and placement among edge servers in the cooperative MEC. However, it still needs to face some similar limitations to the optimization approaches. In other words, the MFG problem has to be reformulated and solved whenever the task distribution changes.

To address the above issues, in this article, we propose an MFG-guided deep RL (DRL) approach for task placement in the cooperative MEC. Different from the existing researches considering the combination of MFG and RL [12], [13], which they apply the pure RL method to solve the problem modeled by MFG or draw lessons from the mean field theory to model the action profile in RL, we provide the explicit MFG solution to guide the RL process. With the guidance of the Nash equilibrium solved by the MFG, the RL agent conducts the learning with guided directions. Based on the evolutionary dynamics in the MFG, the mean field guided Q-value (MFG-Q) can be gained and be used for updating the components in the DRL scheme. The proposed approach has the potential to reduce the average service delay and ensure that energy consumption does not increase significantly in the cooperative MEC architecture. Besides, with the guidance of the Nash equilibrium for the MFG, our DRL approach can obtain the clear learning directions, which effectively reduce task placement decision-making delay and improve users' Quality of Service (QoS). Our salient contributions are summarized as follows.

1) To the best of our knowledge, we are the first one to apply the DRL with the guidance of MFG equilibrium in the cooperative MEC architecture. The task placement decisions can be obtained by solving the MFG and the DRL problem jointly. Briefly, we define an MFG-Q, which replaces the target Q-value with the Nash equilibrium obtained from evolutionary dynamics in the MFG. Based on this MFG-Q, our approach can greatly enhance the efficiency in exploring the environment in the early stages and significantly improve the convergence speed of the learning process.
2) Compared to the existing task placement solution in the cooperative MEC, our MFG-guided DRL approach can handle a large number of users' task placement problems without increasing computational complexity. This is because we ensure the same goal of the MFG and the DRL approach when establishing the system model. Accordingly, our approach can better deal with the dynamic task distribution in practice.
3) We illustrate our proposed model based on the real-world users' distribution data set from the Internet service provider. By using our proposed approach, average service delay can be significantly reduced. The performance evaluation shows the advantages of our MFG-guided DRL task placement approach over the pure DRL or MFG based ones in terms of timely decision making.

The remainder of this article is organized as follows. Section II introduces the related works of the cooperative MEC, RL, and MFG. Section III describes the system model of the task placement problem. Section IV provides a detailed description of the solution based on the RL and evolutionary dynamics in the MFG. The simulation results and performance evaluation are shown in Section V. Finally, the conclusion remarks are made in Section VI.

II. Related Works

A. Cooperative Multiaccess Edge Computing

Due to the highly distributed deployment, each MEC server is endowed with the limited computational and storage resources comparing to remote cloud centers. Thus, processing a large amount of workload locally may result in long queuing delay and degradation in QoS. To solve this issue, cooperation among MEC servers to jointly process the tasks is proposed to improve service efficiency. A strategy of clustering servers for cooperative computing was developed in [14] by modeling as a cooperative game. Ning *et al.* [15] proposed an iterative heuristic MEC resources allocation algorithm to solve the mixed-integer linear programming problem. Liu *et al.* [16] implemented a cooperative game-based scheduling method COOPER-SCHED to guarantee the expected deadline. Wang *et al.* [17], [18] proposed the resource allocation strategy for cellular users. Moreover, a socially motivated cooperative MEC framework was developed where the graph model was related to the social relationship among the devices in [19], while Li *et al.* [20] put forward a novel method to optimize the offloading decision and computational resources allocation based on the DRL. However, most existing works offer solutions based on optimization methods. These methods can tackle the stage-based problem very well, which means they can solve the task placement problem in a fixed time period. When the number of users is large, this approach becomes complicated and time consuming. That limits the application of the above methods in dynamic communication environments. Although pure RL frameworks are suitable for dynamic environments, they still face the problems of unstable training and difficulty in convergence.

B. Deep Reinforcement Learning With Game Theory

RL, as a type of machine learning methods, can efficiently solve the model-free MDP problem. It has been applied in several applications as its good performance recently. Different from the pure RL problem, some scholars have opened up a new way for the development of the RL, jointly considering RL and game theory. Littman [21] described a Q-learning-like algorithm for finding optimal policies in two-player zero-sum stochastic games. Furthermore, Hu and Wellman [22], [23] proposed a general-sum case in stochastic games by learning a Nash Q-value. Moreover, the n-agent general sum game can be transformed into a zero-sum game of two agents by

applying the Friend-or-Foe Q-Learning [24] solution, while Bowling and Veloso [25] applied the dynamic learning rate to speed up the convergence.

With the rapid development of the MFG [9], [26], some works focus on the integration of the MFG and the RL very recently. Yang *et al.* [12] converted the MFG problem to a single agent MDP scheme and employed the inverse RL approach to solve it. Yang *et al.* [13] proposed a new mean field Q-value as an estimation of the optimal Q-value and analyzed the convergence to the Nash equilibrium, and Li *et al.* [27] also applied mean field Q-value to the multicritic one-actor network framework. A method for calculating the optimal strategy in a multiagent system is proposed in [28], which can be applied to a case where the number of agents is extremely large. This article follows the same direction with [12] and [23], we establish the relationship between the MFG and the RL, and find the optimal policy for each stage in the RL. Inspired by the Nash Q-value defined in [23], we consider the MFG problem formulation and define the MFG-Q. Moreover, different from the framework proposed in [12], we consider the specific MFG solution to guide the learning process, rather than just establishing the MFG model and applying the pure RL solution. We propose an MFG-Q as the estimation of the optimal Q-value, which can be gained according to the solution in the MFG problem.

We consider the task placement problem in the cooperative MEC by applying the RL approach with the guidance of MFG. Different from the existing solutions listed above, our method does not increase the computational complexity in the case of an extremely large number of users compared to the traditional optimization methods. Moreover, after the agent learned the MEC environment, the task placement decisions can be obtained very quickly, rather than doing complex calculations each time due to different user distributions.

III. Task Placement System Model

In this section, we establish the system model of the task placement problem in the cooperative MEC. Due to the limited and unbalanced distribution for computational resources of MEC servers, the tasks that end users upload to the MEC servers need to be reassigned and scheduled between the MEC servers to improve the system performance and meet user needs.

A. Task Placement Problem Formulation

Consider a basic cooperative MEC system which consists of multiple end users and multiple MEC servers, and all of the end users (e.g., smartphones, laptops, and Internet of Vehicles) are covered by MEC servers (e.g., small base stations and wireless access points) at their current locations, as shown in Fig. 1. End users have their tasks to deal with, and they will upload the tasks to the MEC server for executing over wireless communication. After completing the tasks, the servers will return the results of the tasks to the end users. Once a user's task is uploaded, the task may be placed to be executed on the current server or an adjacent server by the guidance of the cloud server, which means the task can be transferred among the MEC servers.

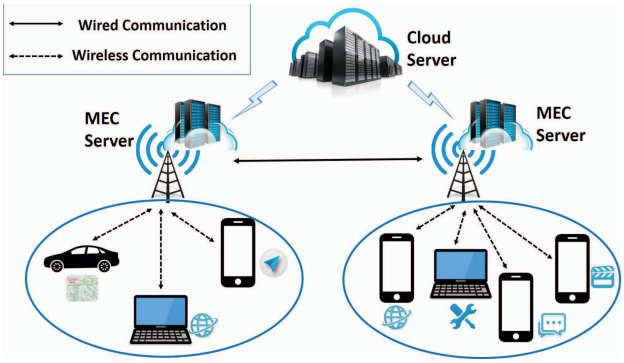


Fig. 1. Cooperative MEC.

We consider a scenario consisting of the number of $\mathcal{D}(j \in \mathcal{D} = 1, 2, \ldots, D)$ MEC servers and the number of $\mathcal{U}(i \in \mathcal{U} = 1, 2, \ldots, U)$ end users covered by servers. Each server is modeled by $M/M/1$ queue. We assume that each user has only one task, so the index of the user equals the index of the task. Different users can run different applications, and different applications may contain different amounts of computations. Furthermore, each task can only be run on one server at one time. In this case, the computational generation rate for user i is defined as λ_i, which means that user i generates λ_i computation units (CPU cycles) per second. Assume that all of the users are generating the number of computations according to a Poisson process with the same mean rate $\bar{\lambda}$. Thus, if the task of user i is executed in MEC server j, the expected service delay in server j can be defined as

$$t_j = \frac{1}{\mu_j - \sum_{i=1}^{U} x_{i,j}\lambda_i} \tag{1}$$

where μ_j is the service rate of MEC server j, and $x_{i,j}$ is defined to describe the connection situation between user i and server j. If the task of user i is executed in server j, $x_{i,j} = 1$; otherwise, it is set to be 0. The time consumption t_{ij} contains both the waiting and execution delay. The denominator is the stable processing speed (the amount of computations executed per second), i.e., the frequency for server j. We can find that the processing speed (frequency) decreases as the tasks load increasing in the server.

Accordingly, the expected energy consumption for the user's task executed in server j is

$$e_j = \kappa_j(\mu_j)^2 \tag{2}$$

where κ_j is defined as the effective capacitance coefficient that depends on the chip architecture at the MEC server. Due to the different computational capacity and CPU architectures of different MEC servers, service rate μ and effective capacitance coefficient κ are different for each MEC server.

Our goal is to reduce the service delay of the entire system by properly placing the servers that handle users' tasks. In the meantime, we do not want to sacrifice too much energy while reducing latency. Therefore, the balance factor ζ is introduced,

and the cost function of users in each server can be defined as

$$l_j = t_j + \zeta e_j. \tag{3}$$

ζ is a factor to balance the two components in the cost function, and e_j can be considered as a penalty. Thus, we have the cost function of the system, which can be defined as the expected service delay and energy consumption per user. We try to reduce the average service delay and do not want to increase too much energy consumption during the process. Therefore, the cost function can be described as

$$\begin{aligned} L &= \frac{1}{U}\sum_{j=1}^{D}\sum_{i=1}^{U} x_{i,j}\lambda_i f_j \\ &= \frac{1}{U}\sum_{j=1}^{D}\sum_{i=1}^{U} \frac{x_{i,j}\lambda_i}{\mu_j - \sum_{i=1}^{U} x_{i,j}\lambda_i} + \frac{1}{U}\zeta\sum_{j=1}^{D}\sum_{i=1}^{U}\kappa_j x_{i,j}\lambda_i\left(\mu_j\right)^2 \\ &= \frac{1}{U}\sum_{j=1}^{D}\frac{N_j\bar{\lambda}}{\mu_j - N_j\bar{\lambda}} + \frac{1}{U}\zeta\sum_{j=1}^{D}\kappa_j N_j\bar{\lambda}\left(\mu_j\right)^2 \\ &= \sum_{j=1}^{D} s_j\frac{\bar{\lambda}}{\mu_j - N_j\bar{\lambda}} + \zeta\sum_{j=1}^{D} s_j\kappa_j\bar{\lambda}\left(\mu_j\right)^2. \end{aligned} \tag{4}$$

U is the number of end users in the system and N_j is the number of users whose tasks are executed in server j. s_j is the proportion of tasks in server j, which is equal to N_j/U. Our object is to minimize the cost function L by scheduling the user's pending task on different servers with the following constraints. The minimization problem can be formulated as

$$\min_{s} \; L = \sum_{j=1}^{D} s_j\frac{\bar{\lambda}}{\mu_j - s_j U\bar{\lambda}} + \zeta\sum_{j=1}^{D} s_j\kappa_j\bar{\lambda}\left(\mu_j\right)^2 \tag{5}$$

$$\text{s.t.} \;\; \lambda_i > 0 \quad \forall i \in U \tag{6}$$

$$\sum_{i=1}^{U} x_{i,j}\lambda_i < \mu_j \quad \forall j \in D. \tag{7}$$

B. Mean Field Game on Graph

The MFG theory can be efficiently applied to the problems where there are a large number of players or agents. More specifically, it can reduce the computational complexity of the problem, especially, when the number of players is extremely large compared with the traditional game theory framework, such as the classical evolutionary game or Nash noncooperative game. The single player can update its strategy through interacting with the collective behavior (i.e., mean field) of other players in the game. Since the MEC can be implemented by a large number of users or tasks in which the individual user or task can have an impact on the whole system, we propose the MFG to formulate the task placement problem as described in Section III-A in the cooperative MEC. The detailed presentation of MFG on the graph is shown as follows.

Each user is an individual agent in the MFG. Assume that all of the MEC servers make up a graph, and the tasks can be scheduled to the adjacent MEC servers from the current server in one scheduling step. Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a strategy graph. In the strategy graph, the vertex set $\mathcal{V} = \{1, 2, \ldots, D\}$ represents all possible states (MEC servers in Section III-A) of the users, which means each task can be allocated to D different servers; the edge set $\mathcal{E}$ consists of all possible transition paths between servers [the task can be scheduled from servers m to n only if $(m, n) \in \mathcal{E}$]. For each vertex m, we define the possible transaction situation as $\mathcal{V}_m \coloneqq \{n : (n, m) \in \mathcal{E}\}$. By considering the current vertex itself, we have $\overline{\mathcal{V}}_m \coloneqq \mathcal{V}_m \cup \{m\}$. This means the task can be placed to the servers $n \in \overline{\mathcal{V}}_m$ on the next time step if the current server is m. Let s_m be the proportion of users in state m, i.e., the proportion of users who are assigned to run the task in server m. Additionally, $\mathbf{s} \coloneqq (s_1, s_2, \ldots, s_D)$ is same as the proportion s in (5). According to the optimal strategy (transition probability) found by MFG, the task distribution s will change to minimize cost L. In this way, the major components of MFG on the graph can be defined as follows.

1) *Population Distribution s^t:* s_m^t represents the fraction of users whose tasks are executed on MEC server m at time step t.
2) *Transition Probability a^t:* a_{mn}^t is the probability that the task in server m is transferred to server n for execution at time t. Furthermore, a_m^t is the *action* of users in server m. a_{mn} is equal to 0 if servers m and n are not connected. The forward Fokker–Planck (FPK) equation for each server n can be defined as
$$s_n^{t+1} = \sum_{m \in \overline{\mathcal{V}}_n} a_{mn}^t s_m^t. \tag{8}$$
3) *Reward $r_m(s^t, a_m^t) \coloneqq -\sum_{n\in\overline{\mathcal{V}}_m} a_{mn}^t r_{mn}(s^t, a_m^t)$:* This is the instant reward received by the users whose tasks are executed on servers m with population distribution s^t and choose action a_m^t. The reward r_{mn} can be defined as the negative of the average cost for users whose tasks are placed from server m to n at time t according to (3), and r_m can be seen as the negative of the expected cost for the users in server m. Moreover, the reward r of all the users in the system is the negative of the cost function $-L$ as shown in (5) at time t.
4) *Value Function V^t:* The value function represents expected cumulative rewards at time t. The value function of each server m is defined as follows, which is also the backward Hamilton–Jacobi–Bellman (HJB) equation in MFG:
$$\begin{aligned} V_m^t &= \max_{a_m^t}\left\{ r_m\left(s^t, a_m^t\right) + \sum_{n\in\overline{\mathcal{V}}_m} a_{mn}^t V_n^{t+1} \right\} \\ &= \max_{a_m^t}\left\{ \sum_{n\in\overline{\mathcal{V}}_m} a_{mn}^t r_{mn}\left(s^t, a_m^t\right) + \sum_{n\in\overline{\mathcal{V}}_m} a_{mn}^t V_n^{t+1} \right\}. \end{aligned} \tag{9}$$

According to the major components shown above, the Nash equilibrium can be obtained as the solution of our MFG on the graph.

Definition 1: In the above description, the Nash equilibrium can be defined as follows with the optimal action a^*:

$$-L\left(s, a^*\right) \geq -L\left(s, \bar{\mathcal{A}}\left(\bar{a^*}\right)\right) \tag{10}$$

$$L\left(s, a^*\right) \leq L\left(s, \bar{\mathcal{A}}\left(\bar{a^*}\right)\right). \tag{11}$$

$\bar{\mathcal{A}}(\bar{a^*})$ is the action space of all possible actions except optimal action a^*. This means at distribution s, the rewards of all users (cost L) with optimal action a^* cannot be decreased by changing to other actions $\bar{a^*} \in \bar{\mathcal{A}}$. The solution of this MFG is to find the optimal action at each step, which is satisfied with FPK (8) and HJB (9).

C. Mean Field Model to Markov Decision Process

In this section, we prove that the above MFG problem on the graph can be converted to the single-agent MDP. State trajectory achieved by a series of actions in the MDP is in accordance with the state dynamics in the MFG, that is, FPK equation in (8). This can lead to the conclusion that the MDP and the MFG have the same goal, and solving the above MFG is equivalent to find the optimal solution for the MDP. For this MFG on the graph, it can be changed to an RL problem that fits within the Markov properties. It is important to note that in the above MFG problem, the generic users obtain their optimal strategy with the information of population density. Therefore, it can be converted to a single-agent RL problem, which means our proposed task placement system is a central assisted. We will prove that the above MFG problem can be converted to a single-agent RL model in the following parts of this section. The definition of the RL problem is given as follows.

1) *State $s^t \in \mathcal{S}$:* The distribution of all the users at time t

$$s^t = \left(s_1^t, s_2^t, \ldots, s_D^t\right). \tag{12}$$

D is the number of MEC servers.

2) *Action $a^t \in \mathcal{A}$:* The transaction probability at time t. The action is the proportion of the users whose tasks in the current server switching to other servers (include the current server itself) at the next moment. It can represent the flow of tasks among the servers

$$a^t = \left(a_1^t, a_2^t, \ldots, a_N^t\right). \tag{13}$$

There will be N strategies in each action at time t. We redefine the index of each strategy for a more convenient way to explain the physical meaning of the action. It is related to the shape of the graph and the connection relationship between the servers. Specifically, N is equal to the number of undirected connection paths (edges) in the graph. For instance, if D servers are linearly arranged, which means each server has two adjacent servers. In this case, N is equal to $D-1$, i.e., there are N kinds of connections (edges) in our graph.

3) *Reward $r(s^t, a^t)$:* It is the reward received by all the users with the state s^t and action a^t, which is equal to the negative of the cost $-L$ at time t. Our goal is to maximize the reward, that is, minimize the cost L

$$r\left(s^t, a^t\right) = -\sum_{m=1}^{D} s_m^t \sum_{n \in \overline{\mathcal{V}}_m} a_{mn}^t r_{mn}\left(s^t, a_m^t\right) = -L^t. \tag{14}$$

We now define $V^*(s^t)$ as the weighted sum of all the value functions defined in MFG for the servers. Similar to [12], we show that it is the value function of the above MDP problem, which will be satisfied with the Bellman optimally equation. Once $V^*(s^t)$ can be proved to satisfy the Bellman optimally equation, we can say that the proposed MFG problem can be converted to the above single-agent MDP. The proof can be described as follows:

$$V^*\left(s^t\right) = \sum_{m=1}^{D} s_m^t V_m^t \tag{15}$$

$$= \sum_{m=1}^{D} s_m^t \max_{a_m^t} \left\{ \sum_{n \in \overline{\mathcal{V}}_m} a_{mn}^t r_{mn}\left(s^t, a_m^t\right) + \sum_{n \in \overline{\mathcal{V}}_m} a_{mn}^t V_n^{t+1} \right\} \tag{16}$$

$$= \max_{a^t} \left\{ \sum_{m=1}^{D} s_m^t r_m\left(s^t, a_m^t\right) + \sum_{m=1}^{D} \sum_{n=1}^{D} a_{mn}^t s_m^t V_n^{t+1} \right\} \tag{17}$$

$$= \max_{a^t} \left\{ r\left(s^t, a^t\right) + \sum_{n=1}^{D} s_n^{t+1} V_n^{t+1} \right\} \tag{18}$$

$$= \max_{a^t} \left\{ r\left(s^t, a^t\right) + V^*\left(s^{t+1}\right) \right\}. \tag{19}$$

Therefore, from the above proof, $V^*(s^t)$ satisfies the Bellman optimally equation for the single-agent MDP problem defined in this section. It should be noted that the above MFG on the graph model cannot be converted to a multiagent MDP. In multiagent MDP, the value function for one agent is defined only in terms of itself, but the value function for the agent in an MFG explicitly depends on the value functions of other agents [12]. As a result, the state dynamics with the optimal policy solved by the MDP is equivalent to the state trajectory obtained by the Nash equilibrium in MFG. More specifically, at state s, the actions a under the optimal policy in the MDP are the same as the optimal policy a^* with the Nash equilibrium in the MFG.

IV. Mean Field Game Guided Deep Reinforcement Learning Approach

In this section, an approach combining the RL and the MFG is proposed. More specifically, the RL approach that does not need to know the state transition function is given to solve the MDP. Then, MFG-Q that can be obtained by leveraging the evolutionary dynamics in the MFG is defined to solve the task placement problem in the cooperative MEC.

A. Reinforcement Learning With Nash Equilibrium

First, the general RL problems with the Nash equilibrium in game theory are introduced. Subsequently, MFG-Q is defined based on the Nash equilibrium strategies gained in MFG.

In the RL or the MDP problem, with the policy $\pi := \{a^1, a^2, \ldots, a^t\}$, the agent will take action a in state s at each time step t. For one stage, policy π is a probability of choosing an action, which can be represented as $\pi(a|s) \in [0, 1]$. After taking the action a, the agent will move to the next state s' and receive the reward r. The agent's object is to maximize the state value function, which can be written as the sum of

the expected discounted rewards

$$V_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[r^t | \pi, s_0 = s\right] \tag{20}$$

where $\gamma \in (0, 1]$ is the discount factor, and s_0 is the initial state. It represents the value of state s under the policy π.

For the Q-learning algorithm, the state–action value (Q-value) can be defined as the value function, which follows the Bellman equation. The optimal Q-value which is the sum of the expected discounted reward by following the optimal policy π^*, which can be represented as:

$$Q^*(s, a) = r(s, a) + \gamma \mathbb{E}_{s'}\left[V_{\pi^*}(s')\right] \tag{21}$$

where $r(s, a)$ is the reward for taking action a at state s and s' is the state of the next time step. The value function can be expressed as the expected of the Q-values in (21), which is

$$V_\pi(s) = \mathbb{E}_a[Q_\pi(s, a)], \quad V_{\pi^*}(s) = \mathbb{E}_a\left[Q^*(s, a)\right]. \tag{22}$$

The optimal policy π^* can be found by applying value-based frameworks [such as Q-learning and deep Q-network (DQN)] or policy-based frameworks (such as the actor–critic approach) if we know the optimal Q-value $Q^*(s, a)$. The definition of optimal Q-value establishes a relationship between the current time step and the next time step. Moreover, the action is the deterministic policy not the stochastic policy in the Q-learning algorithm. Hence, the optimal state value can be redefined as $V_{\pi^*}(s) = \max_a Q(s, a)$. According to this property, Q-learning initializes its Q-value, and update the Q-value by the following equation:

$$\begin{aligned} Q^{t+1}(s, a) &= (1 - \alpha) Q^t(s, a) + \alpha \left[r^t + \gamma \max_{a'} Q^t(s', a') \right] \\ &= Q^t(s, a) + \alpha \left[r^t + \gamma \max_{a'} Q^t(s', a') - Q^t(s, a) \right] \end{aligned} \tag{23}$$

where α is the learning rate. $r^t + \gamma \max_{a'} Q^t(s', a')$ is known as the temporal difference (TD) target, where $\max_{a'} Q^t(s', a')$ is called the target Q-value. It takes the difference between the actual and predicted values, and we call it the TD error. Moreover, it can be proved that the Q-value will converge to the optimal Q-value if all the states and actions can be visited infinitely.

Combining game theory with the RL framework, the optimal policy π^* can be seen as the Nash equilibrium in game theory. Therefore, the Nash equilibrium is the policies π^*, which can be defined as

$$V_{\pi^*}(s) \geq V_\pi(s) \text{ for all } \pi \in \Pi \tag{24}$$

where Π is the set of all possible policy in state s.

By considering the Nash equilibrium in game theory. Hu and Wellman [22] proposed a new optimal Q-value estimation method, where they replace the target Q-value with Nash equilibrium solved by the game theory at each stage. This update estimation is called Nash Q-value, which is defined by the following equation:

$$\text{Nash } Q^*(s, a) = r(s, a) + \gamma \mathbb{E}_{s'}\left[\text{Nash } V_{\pi^*}(s')\right]. \tag{25}$$

The optimal policy π^* is the Nash equilibrium which can be gained through the game theory solution. Hu *et al.* [22] also proved the convergence of the Q-value to an equilibrium Nash Q-value.

Similar to the Nash Q-value, we define the MFG-Q $Q_{\text{MF}}(s, a)$ which can be obtained by solving the MFG problem. We have both the MFG and RL approach in our scheme. In the learning process, the agent takes the action a, gets the reward r and moves to the new state s'. After that, the mean field Nash equilibrium $\pi^*(s')$ is calculated according to the MFG for computing the mean field guided state value ($V_{\text{MF}}(s')$) in the RL approach. Based on this mean field guided state value ($V_{\text{MF}}(s')$), the learning agent will select the action according to the RL approach. Thus, the new update rule which considers the mean field Nash equilibrium for the Q-value and TD error can be defined as

$$Q^{t+1}(s, a) = (1 - \alpha) Q^t(s, a) + \alpha \left[r^t + \gamma V_{\text{MF}}{}^t(s') \right] \tag{26}$$

where

$$V_{\text{MF}}(s') = \mathbb{E}_{a'}\left[\pi^* Q_{\text{MF}}(s', a')\right] \tag{27}$$

$$= \sum_{a'} \pi^*(a'|s') Q_{\text{MF}}(s', a'). \tag{28}$$

In each time step, the agent will update the Q-value according to (26) and (28). In this way, the key target of the RL problem is how to find the optimal policy π^*, which can be seen as the learning guide of the agent. Since optimal policy π is the mean field equilibrium point in the game theory, we will provide a solution of how to get this equilibrium in the next section.

Like (21) and (25), we define the MFG-Q $Q_{\text{MF}}(s, a)$ in the form of

$$Q_{\text{MF}}(s, a) = r(s, a) + \gamma \mathbb{E}_{s'}\left[V_{\text{MF}}(s')\right]. \tag{29}$$

It is also an improvement of the estimation for the optimal Q-value. Different from the general RL problem where the estimation of the optimal Q-value is gained according to the existing action patterns, our MFG-Q is gained by assuming the equilibrium the agent can reach with the learning environment. By the guidance obtained from the MFG, the direction for updating the Q-value is more clear. Therefore, the convergence rates of the learning process will be significantly improved. More specifically, our MFG-Q is more suitable for the problems that follow the mean field property, i.e., the model can be formulated with a large number of populations, such as the task placement in this article.

B. Mean Field Evolutionary Dynamics Approach

Section IV-A provides the framework for the DRL solution. Moreover, it needs the Nash equilibrium obtained by the MFG solution. Thus, the problem is converted to find the optimal policy based on the MFG solution for the RL in each stage. In this section, based on the problem formulation in Section III-B, we leverage evolutionary dynamics in the MFG to find the Nash equilibrium.

In Section III-B, each user's task will be assigned to one of the servers to execute. Thus, the population distribution,

which is the strategy of users in the MFG can be defined as

$$\mathcal{P}(\mathcal{S}) = \left\{ (s_j)_{j=1}^{D} : \sum_{j=1}^{D} s_j = 1, \quad s_j \geq 0, \ j \in \mathcal{D} \right\} \tag{30}$$

where s_j is the proportion of users' tasks placed in server j. Based on the knowledge of the population game and the MFG, the mean field evolutionary dynamics (MFED) [29] is defined by the following theorem.

Theorem 1: Suppose that a population MFG on the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the constant $\beta \geq 0$ and the payoff function $F : \mathcal{P}(\mathcal{S}) \to \mathbb{R}^n$ are continuous. The evolutionary dynamics is given by

$$\begin{aligned} \frac{\mathrm{d}s_j}{\mathrm{d}t} = & \sum_{n \in \overline{\mathcal{V}}_j} \frac{1}{d_n} s_n \left[F_j(s) - F_n(s) + \beta \left(\log s_n - \log s_j \right) \right]^+ \\ & - \sum_{n \in \overline{\mathcal{V}}_j} \frac{1}{d_j} s_j \left[F_n(s) - F_j(s) + \beta \left(\log s_j - \log s_n \right) \right]^+ \end{aligned} \tag{31}$$

where $\beta \geq 0$ is the strength of uncertainty, $[\,\cdot\,]^+ = \max\{\cdot, 0\}$. Same as the definition in Section III-B, n is the index of the neighboring servers of server j. $d_j = \sum_{n \in \overline{\mathcal{V}}_j} 1$ is the number of the neighboring servers, which represents the degree of graph on vertex j. It can be seen as the FPK equation in MFG, which is also the dynamics of the users' distribution.

Considering the scenario of our task placement in the cooperative MEC problem, we regard the potential as L which is given in (5). Accordingly, the payoff function F of the potential L can be defined as

$$F(j) = \frac{\bar{\lambda} \mu_j}{\left(\mu_j - s_j U \bar{\lambda} \right)^2} + \zeta \kappa_j \bar{\lambda} \left(\mu_j \right)^2 \quad \forall j \in D. \tag{32}$$

Chow *et al.* [29] proved that minimizing each player's payoff is equivalent to minimizing the potential. As the MFG model described in Section III-B. Our goal is to achieve a better state of user distribution s by finding an optimal action (transaction probability a), which can minimize the cost L. To obtain the optimal distribution s, a gradient matrix $\Lambda \in \mathbb{R}^{D \times D}$ is constructed to identify the action a, which helps the users make up the new distribution.

We consider the scenario that MEC servers are arranged on the side of the road. In this situation, the MEC servers are linearly arranged, i.e., the tasks in the current server can only be placed to the neighboring two servers in the next time step. Thus, our strategy graph is a one-dimension graph, and each vertex node (server) only has two edges (connection relationship) to the adjacent servers. In each time step, the distribution of users will change based on the task placement in adjacent servers. We define that the gradient matrix $\Lambda \in \mathbb{R}^{D \times D}$ is a population flow matrix, and Λdt represents the proportion of the flows, which can be seen as the action a as described in Section III-B, where dt is the size of the step. For example, $a_{mn} = \Lambda(m, n)dt = p$ means that p percent of the tasks in server m will be placed to server n according to our strategy.

More specifically, through Theorem 1, gradient matrix $\Lambda \in \mathbb{R}^{D \times D}$ will be computed according to the following equations. Due to server 1 and server D are only connected to one neighboring server, the gradients related to them can be given as

$$\begin{aligned} \Lambda(1,1) &= -[F(1) - F(2)]^+ \\ \Lambda(1,2) &= [F(2) - F(1)]^+ \\ \Lambda(D,D) &= -[F(D) - F(D-1)]^+ \\ \Lambda(D,D-1) &= [F(D-1) - F(D)]^+. \end{aligned} \tag{33}$$

For the other servers $j : \{j \in \mathcal{D} : j \neq 1, D\}$, they have connections to server $j-1$ and $j+1$. Therefore, the related gradients can be computed as

$$\begin{aligned} \Lambda(j,j) &= -\left[F(j) - F(j+1)\right]^+ - \left[F(j) - F(j-1)\right]^+ \\ \Lambda(j,j+1) &= \left[F(j+1) - F(j)\right]^+ \\ \Lambda(j,j-1) &= \left[F(j-1) - F(j)\right]^+. \end{aligned} \tag{34}$$

During iteration t, the new distribution s^t is updated by distribution s^{t-1} from the previous iteration and the gradient matrix Λ in the following way:

$$s^t = s^{t-1} + \Lambda^t s^{t-1} dt. \tag{35}$$

This solution is called the MFED solution. Back to our RL problem described in Section IV-A, the MFG-Q $Q_{\mathrm{MF}}(s, a)$ can be obtained from this MFG solution. In (27), s' is the new distribution we get, and a' can be represented as $\Lambda^t dt$.

C. Implementation and Overall Architecture

From the perspective of the RL approach, due to the high dimension of action space (the task placement strategies are unlimited) and state space (users' distribution status are unlimited) for our task placement in the cooperative MEC, the DRL framework is considered to solve the dimensional explosion problem. We exploit a new DRL approach with the deep deterministic policy gradient (DDPG) [30] approach, an actor–critic scheme as our foundation. The framework of our MFG-guided DRL approach is shown in Fig. 2. In one state, the servers interact with the MEC environment (transferring the users' tasks among the servers) and get the instant reward. After that, the critic network calculates the TD error according to the instant reward and the optimal action through the MFG

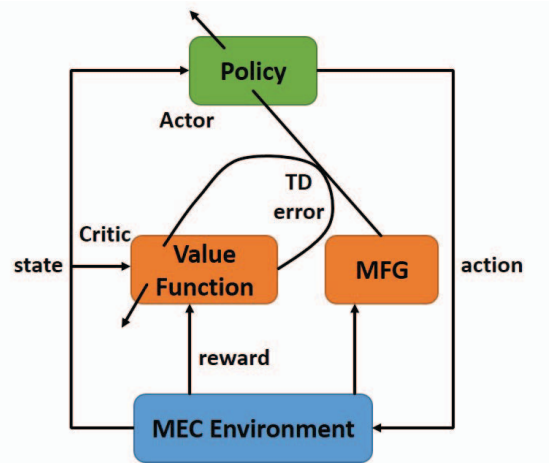


Fig. 2. Framework of the proposed MFG-guided DRL approach.

solution. The MFG solution can also be obtained from the existing MEC environment. This TD error guides the actor network to update its parameters and generate a new action.

In the critic network, we apply the neural network with the parameter ω as the function approximator to represent the Q-value. The updated rule in (26) can be reformulated with the parameter ω. Similar to the DQN [31], the *Loss* function of the critic network with the MFG-Q can be defined as

$$\mathbb{L} = \left(y - Q^{\omega}(s, a)\right)^2 \tag{36}$$

where $y = r(s, a) + \gamma V_{\mathrm{MF}}{}^{\omega'}(s')$ is the target mean field value with the parameter ω'. We can minimize the *Loss* function by applying the gradient-based optimization method, i.e., mini-batch stochastic gradient descent.

Different from the value-based framework, we can obtain the policy from the actor network $\Omega(s)$ with parameter θ directly. The target of the actor network is to find the policy which can maximize the objective function J. The actor network is trained by using sampled policy gradient, and the gradient can be calculated as

$$\nabla_{\theta} J(\Omega) = \nabla_a Q^{\omega}(s, a)|_{a=\Omega(s)} \nabla_{\theta} \Omega^{\theta}(s). \tag{37}$$

Different from DDPG, we only have one actor network, which means we do not have the target actor network. We do not need the target action from the target actor network to update the critic network. We directly apply the TD error with the Nash equilibrium solved by the MFG as described in (26) to update the actor and critic networks.

The pseudocode for our MFG-guided DRL solution is manifested in Algorithm 1. First, the users' population interacts with the MEC environment from step 1 to step 6. Moreover, we introduce the noise process $\mathcal{N}$ to balance the exploration and explanation, and $\mathcal{N}$ is the Gaussian noise. In step 9, we set the label of critic network according to (28) with the parameter ω'. Furthermore, the optimal policy for calculating label can be gained by solving the (35), which is related to the solution of the MFG. In this situation, different from the DDPG method, the process of calculating TD error in our approach contains the contribution of MFG as shown in Fig. 2. Parameters updating for the critic and actor network are shown in steps 10 and 11. Finally, the parameters of the target network Q' are updated by copying from the critic network Q according to a "soft" way [30]. τ is the soft parameter, and this can make the target value change slowly.

Algorithm 1 MFG-Guided DRL Algorithm

Initialization: Randomly initialize the critic network $Q^{\omega}(s, a)$ and actor network $\Omega^{\theta}(s)$ with parameter ω and θ separately. Initialize the replay memory $\mathbb{M}$ and set the parameters of target network Q' with the parameter ω' by copying from Q.

Output: the selected action for all of the tasks.

1: **for** episode =1, ..., N **do**
2: Initialize the noise process $\mathcal{N}$ for action exploration.
3: The users' distribution at the state s_1.
4: **for** $t = 1, ..., T$ **do**
5: Select action $a_t = \Omega^{\theta}(s^t) + \mathcal{N}_t$ according to the actor network and the noise process.
6: The population executes the action a^t, gets the immediate reward r^t, and observes the next state s^{t+1}.
7: Store transition (s^t, a^t, r^t, s^{t+1}) in memory.
8: Randomly sample mini-batch size M of transitions (s_j, a_j, r_j, s_{j+1}) in memory.
9: Set Q value as $y_j = r_j + \gamma V_{\mathrm{MF}}{}^{\omega'}(s_{j+1})$ by (28), where the optimal policy can be obtained by (35).
10: Update the Critic network by minimizing the Loss function $\mathbb{L} = \frac{1}{M}\sum_j (y_j - Q^{\omega}(s_j, a_j))^2$ according to Adam Algorithm.
11: Update the actor network by sampled policy gradient with $\nabla_{\theta} J(\Omega) = \frac{1}{M}\sum_j \nabla_a Q^{\omega}(s, a)|_{a=\Omega(s_j)} \nabla_{\theta} \Omega^{\theta}(s_j)$.
12: Update the parameter ω' in target networks Q' in "soft" way: $\omega' \leftarrow \tau\omega + (1-\tau)\omega'$
13: **end for**
14: **end for**

V. Performance Evaluation

To ascertain the performance of our proposed method, we perform the following comprehensive simulations. We consider the problem that MEC servers are linearly arranged, and each server is connected to two adjacent servers. This problem is very applicable to the scene of the base station arranged along the road. Accordingly, we obtain the real-world user distribution along the road from 4G eNBs data set of the Internet service provider. The data set contains network traffic data collected from 3072 4G eNBs in a southern city of China in 2015.

We select one month period and obtain the users' distribution at different times from the data set as our initial users' distributions. At this time, the performance of the task placement with the initial users distribution can be seen as the baseline. All of the users are generating the computations according to the Poisson process with the mean rate $\bar{\lambda} = 3 \times 10^7$ cycle. Assume that there are around 1×10^4 users, and we place ten MEC servers along the road for service where the service rate for each MEC server belongs to $\mu \in (300, 400)$ GHz. We set the effective capacitance coefficient κ_j belongs to $(0.8, 1.2)$ (10^{-20} J·s^2/cycle^3) due to the different CPU architectures of servers. In our implementation, we adopt the fully connected neural networks with two hidden layers for our actor and critic networks, where each hidden layer has 128 nodes. For the neural network training process, we set the experience replay memory size as 2000, the batch size as 32, and the learning rate as 0.001. For the RL part, discount factor γ is 0.9 and soft parameter τ is 0.01. Particularly, all of the parameters of the learning approach are got from parameter tuning.

Our goal is to reduce the average service delay. In the process of reducing the delay, the energy consumption will increase. Therefore, we want to ensure that energy consumption does not increase too much according to our approach. The balance factor ζ in (5) is adjusted to achieve this goal. To choose the best value of the balance factor ζ, we evaluate the

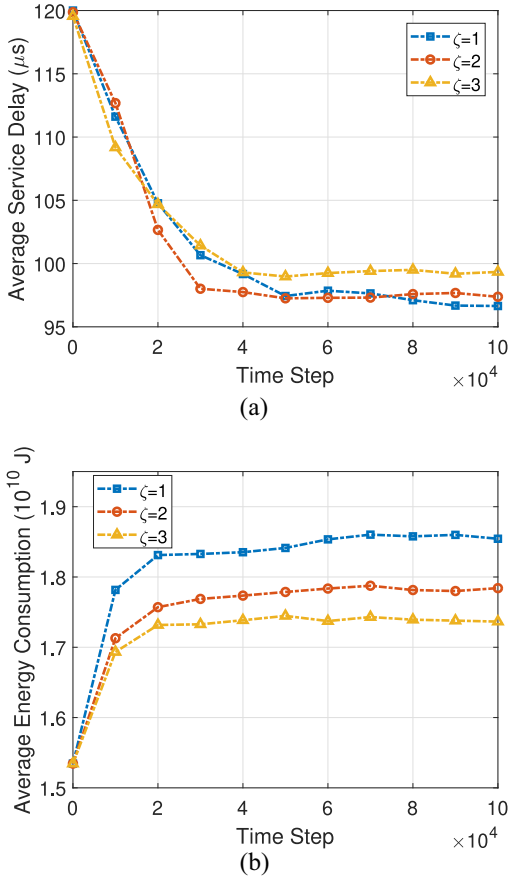


Fig. 3. Adjust balance factor ζ. (a) Service delay reduction. (b) Energy consumption enhancing.

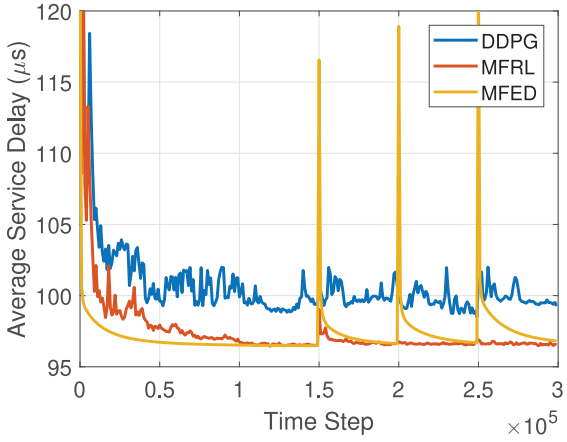


Fig. 4. Compare with the DRL and MFG method.

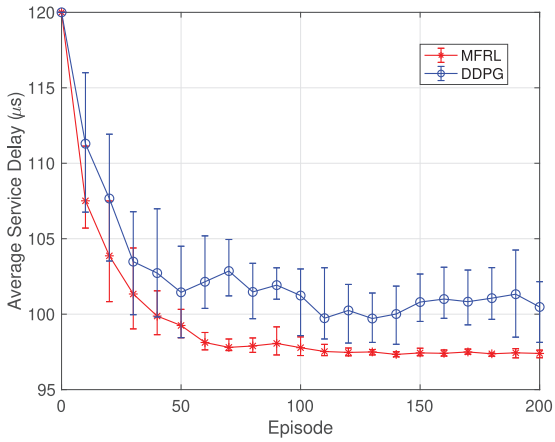


Fig. 5. Compare MFRL with DDPG in the dynamic environment.

performance during changing the different values of ζ under a fixed-time user initial distribution. We set ζ as $1, 2, 3 \times 10^{-8}$ to ensure that delay and energy are on the same scale. To avoid the high variance of the learning process, we take five experiments and the average evaluation results are shown in Fig. 3. Each step represents one iteration. We find that when ζ is equal to 2, the service delay in Fig. 3(a) reduces significantly, and the energy consumption in Fig. 3(b) does not increase too much. When ζ is equal to 1, the delay reduction 19.16% is very similar to 18.75% when $\zeta = 2$, but the increase in energy consumption is huge, and this imbalance also happens when $\zeta = 3$. Therefore, we select $\zeta = 2$ for further evaluation.

We compare our MFG-guided DRL approach with other approaches to verify the effectiveness of our proposed scheme. To facilitate display performance, we temporarily ignore the penalty term "energy consumption" and only show the performance of service delay. The evaluation results are shown in Fig. 4. We mainly focus on the service delay reduction. Comparing to the pure DRL method DDPG and pure MFG solution [32] MFED, our approach (MFRL) has faster convergence rates and less decision-making delay. We show four stages of the training process in Fig. 4, that is, at time step $(0, 1.5, 2, 2.5) \times 10^5$, we have the new user distribution to do the task placement. Different from the DDPG which is not stable and cannot converge to the optimal solution (MFED solution), our MFRL approach converges quickly and approaches the optimal value easily. Even though at the initial stage of the training, our approach still converges quickly, which can speed up the learning process effectively. Besides, the MFED (or other optimization approaches) can only solve the problem when the initial distribution is certain. As shown by the yellow line in Fig. 4, MFED needs to resolve the problem from scratch when the new distribution is coming. However, our approach has the memory of the existing communication environment. When the stage comes to the new distribution, the learning agent can make the decision quickly, and will not lead to intolerable decision-making delay. At the same time, our MFRL approach can converge to the point which is very similar to the MFED solution.

One of our main contributions is that the proposed MFRL approach can converge faster and more stable than the pure DRL approach DDPG. We implement these two approaches in the dynamic environments, which means we provide the task placement strategy for different user distributions over time, and the comparison results are shown in Fig. 5. Each episode represents one new user distribution, and we train 1000 steps (iterations) for each episode. We take the experiments ten times and the average service delay and variance bound are shown in Fig. 5. We find that our proposed MFRL approach can converge only after 50 episodes, but the DDPG

TABLE I
COMPARISON OF SERVICE DELAY AND ENERGY CONSUMPTION

	Initial	DDPG	MFRL	MFED
Service Delay(μs)	123.53	100.32	97.42	97.15
RRD	–	18.79%	**21.14%**	21.35%
Energy Cost (10^8J)	149.83	179.21	176.03	172.12
IRE	–	19.61%	17.49%	14.88%
Reward	-153.50	-136.16	-132.62	-131.57
IRR	–	11.30%	**13.60%**	14.29%

method does not converge until after 100 episodes. Moreover, the variance bound of MFRL is decreasing with the increasing of the episodes, and the variance is almost zero after the convergence. Thus, comparing to the DDPG, the proposed MFRL can achieve a lower service delay with a very small variance. This means our training process is stable and effective.

We implement our MFG-guided DRL approach for the dynamic MEC environment. We calculate the average results, and the comparison is shown in Table I. The reward is the integration of service delay and energy consumption according to (5) with $\zeta = 2$. "RRD" represents the "reduction rate of delay," "IRE" represents the "improvement rate of energy," and "IRR" represents the "improvement rate of reward." "Initial" can be seen as the baseline, that is, tasks have not been assigned among servers. By comparing the service delay reduction, we find that our MFRL approach has much delay reduction than the DDPG approach, and the reduction rate is very close to the optimal results (MFED results). More specifically, compared to the initial state, our proposed MFRL approach reduces the service delay by 21.14%, and it is very close to the MFED results (21.35%). Furthermore, when comparing the improvement of reward, our proposed MFRL method (13.60%) is much better than the DDPG method (11.30%), and it is very close to the MFED method (14.29%). Therefore, our proposed MFRL approach can achieve a similar performance as the optimal MFED approach, and adapt to the dynamic user distribution environment at the same time. When compared with the DDPG approach, service delay reduces more and energy consumption increases less in our proposed MFRL approach.

In summary, our MFG-guided DRL approach can significantly reduce the average service delay for cooperative MEC. Besides, both the learning performance and convergence rates of ours are better than the pure DRL approach. At the same time, the MEC server can make timely task placement decisions based on our approach. This can avoid the unbearable decision-making delay caused by the MFG (optimization) method, which needs to resolve the problem when the distribution of the tasks changing.

VI. CONCLUSION

To reduce the service delay in the cooperative MEC, we have developed an MFG-guided DRL approach to do the task placement. We have ensured that the MFG and the DRL approach had the common objective. Furthermore, based on the mean field guided Q-value (MFG-Q) obtained from the Nash equilibrium in the MFG, the learning directions were clearer of our approach compared to the pure DRL approach. The multiple simulations have shown that our MFG-guided DRL approach was able to reduce service delay by more than 20% and had a significant performance improvement over pure MFG or pure DRL approach. On the one hand, in comparison with the pure DRL approach, apart from reducing much service delay, our approach was more stable and had the faster convergence rates during learning. On the other hand, different from the MFG where we need to resolve the problem for each tasks distribution, our approach could make task placement decisions directly when training is enough. Consequently, we could make the task placement decision timely.

REFERENCES

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[2] L. Li, Y. Li, and R. Hou, "A novel mobile edge computing-based architecture for future cellular vehicular networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, San Francisco, CA, USA, Mar. 2017, pp. 1–6.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[4] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[5] *Openai Five*, OpenAI, San Francisco, CA, USA. Accessed: Jul. 4, 2019. [Online]. Available: https://openai.com/blog/openai-five/

[6] D. Shi *et al.*, "Deep *Q*-network based route scheduling for TNC vehicles with passengers' location differential privacy," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7681–7692, Oct. 2019.

[7] D. Shi, X. Li, M. Li, J. Wang, P. Li, and M. Pan, "Optimal transportation network company vehicle dispatching via deep deterministic policy gradient," in *Proc. Int. Conf. Wireless Algorithms Syst. Appl.*, Hawaii, USA, Jun. 2019, pp. 297–309.

[8] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3072–3108, 4th Quart., 2019.

[9] J.-M. Lasry and P.-L. Lions, "Mean field games," *Jpn. J. Math.*, vol. 2, no. 1, pp. 229–260, Mar. 2007.

[10] Q. Cheng *et al.*, "Beam-steering optimization in multi-UAVs mmWave networks: A mean field game approach," in *Proc. 11th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Xi'an, China, 2019, pp. 1–5.

[11] Y. Zhang, C. Yang, J. Li, and Z. Han, "Distributed interference-aware traffic offloading and power control in ultra-dense networks: Mean field game with dominating player," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8814–8826, Sep. 2019.

[12] J. Yang, X. Ye, R. Trivedi, H. Xu, and H. Zha, "Learning deep mean field games for modeling large population behavior," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, May 2018, pp. 1–15.

[13] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, Jul. 2018, pp. 1–16.

[14] S. M. S. Tanzil, O. N. Gharehshiran, and V. Krishnamurthy, "Femtocloud formation: A coalitional game-theoretic approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, Dec. 2015, pp. 1–6.

[15] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

[16] C. Liu, K. Li, J. Liang, and K. Li, "COOPER-SCHED: A cooperative scheduling framework for mobile edge computing with expected deadline guarantee," *IEEE Trans. Parallel Distrib. Syst.*, early access, Jun. 7, 2019, doi: 10.1109/TPDS.2019.2921761.

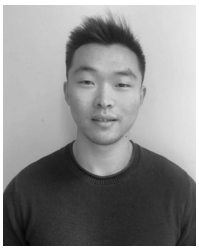
[17] L. Wang, Y. Ai, N. Liu, and A. Fei, "User association and resource allocation in full-duplex relay aided NOMA systems," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10580–10596, Dec. 2019.

[18] L. Wang, H. Tang, H. Wu, and G. L. Stüber, "Resource allocation for D2D communications underlay in rayleigh fading channels," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1159–1170, Feb. 2017.

[19] X. Chen, Z. Zhou, W. Wu, D. Wu, and J. Zhang, "Socially-motivated cooperative mobile edge computing," *IEEE Netw.*, vol. 32, no. 6, pp. 177–183, Nov./Dec. 2018.
[20] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Barcelona, Spain, Apr. 2018. pp. 1–6.
[21] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, New Brunswick, NJ, USA, Jul. 1994, pp. 157–163.
[22] J. Hu and M. P. Wellman, "Multiagent reinforcement learning: Theoretical framework and an algorithm," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Madison, WI, USA, Jul. 1998. pp. 242–250.
[23] J. Hu and M. P. Wellman, "Nash *Q*-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Nov. 2003.
[24] M. L. Littman, "Friend-or-Foe *Q*-learning in general-sum games," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Williamstown, MA, USA, Jul. 2001, pp. 322–328.
[25] M. Bowling and M. Veloso, "Rational and convergent learning in stochastic games," in *Proc. 17th Int. Joint Conf. Artif. Intell. (IJCAI)*, Seattle, WA, USA, Aug. 2001, pp. 1021–1026.
[26] A. Briani and P. Cardaliaguet, "Stable solutions in potential mean field game systems," *Nonlinear Differ. Equ. Appl. NoDEA*, vol. 25, no. 1, pp. 1–26, Feb. 2018.
[27] M. Li *et al.*, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *Proc. World Wide Web Conf.*, San Francisco, CA, USA, May 2019, pp. 983–994.
[28] D. Mguni, J. Jennings, and E. M. de Cote, "Decentralised learning in systems with many, many strategic agents," in *Proc. 32nd AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, Feb. 2018, pp. 1–8.
[29] S.-N. Chow, W. Li, J. Lu, and H. Zhou, "Population games and discrete optimal transport," *J. Nonlinear Sci.*, vol. 29, no. 3, pp. 871–896, Oct. 2019.
[30] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: arXiv:1509.02971.
[31] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[32] H. Gao, W. Li, R. A. Banez, Z. Han, and H. V. Poor, "Mean field evolutionary dynamics in ultra dense mobile edge computing systems," in *Proc. IEEE Global Commun. Conf. (Globecom)*, vols. 1–6. Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

Dian Shi (Student Member, IEEE) received the B.S. degree from the School of Electronic Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2017. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA.

His research interests include deep reinforcement learning, differential privacy, and object detection via Wi-Fi signal.



Hao Gao (Student Member, IEEE) received the B.E. degree in electrical and information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2018. He is currently pursuing the Ph.D. degree in electrical engineering with the University of Houston, Houston, TX, USA.

His current research interests include mean field game and related applications in wireless communication.

Li Wang (Senior Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2009.

She is currently a Full Professor with the School of Electronic Engineering, BUPT, where she heads the High Performance Computing and Networking Laboratory. Her current research interests include wireless communications, distributed networking and storage, vehicular communications, social networks, and edge AI.

Miao Pan (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from Dalian University of Technology, Dalian, China, in 2004, the M.A.Sc. degree in electrical and computer engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2007, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2012.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA. His research interests include cybersecurity, deep learning privacy, big data privacy, cyber-physical systems, and cognitive radio networks.

Zhu Han (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland at College Park, College Park, MD, USA, in 1999 and 2003, respectively.

He is currently a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department and the Computer Science Department, University of Houston, Houston, TX, USA. He is also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul, South Korea. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid.

Dr. Han was an IEEE Communications Society Distinguished Lecturer from 2015 to 2018. He has been an AAAS fellow since 2019 and ACM Distinguished Member since 2019. He has been 1% Highly Cited Researcher according to Web of Science since 2017.

H. Vincent Poor (Life Fellow, IEEE) received the Ph.D. degree in EECS from Princeton University, Princeton, NJ, USA, in 1977.

From 1977 to 1990, he was on the faculty with the University of Illinois at Urbana–Champaign, Champaign, IL, USA. Since 1990, he has been on the faculty with Princeton University, where he is the Michael Henry Strater University Professor of electrical engineering. From 2006 to 2016, he also served as the Dean of Princeton's School of Engineering and Applied Science. His research interests are in the areas of information theory, signal processing, and machine learning, and their applications in wireless networks, energy systems, and related fields. Among his publications in these areas is the forthcoming book *Advanced Data Analytics for Power Systems* (Cambridge University Press, 2020).

Dr. Poor was a recipient of several awards, including the IEEE Alexander Graham Bell Medal in 2017, the ASEE Benjamin Garver Lamme Award in 2019, the D.Sc. Honoris Causa from Syracuse University, awarded in 2017, and the D.Eng. Honoris Causa from the University of Waterloo, awarded in 2019. He is a member of the National Academy of Engineering and the National Academy of Sciences, and a Foreign Member of the Chinese Academy of Sciences, the Royal Society, and other national and international academies.